Performance Model for a Prioritized Multiple-Bus Multiprocessor System

Lizy Kurian
Computer Science and Engineering
University of South Florida
Tampa, FL 33620

Yu-cheng Liu
Department of Electrical Engineering
University of Texas at El Paso
El Paso, TX 79968

Abstract

The performance of a shared memory multiprocessor system with a multiple-bus interconnection network is studied in this paper. The effect of bus and memory contention is modeled. An analytical model to evaluate the acceptance probability of each processor in such a system is presented. It is assumed that each processor in the system has a distinct priority assigned to it and that arbitration is based on priority. Whenever a request from a processor is rejected due to bus or memory conflicts, the request is resubmitted until granted. Effective memory bandwidth of the system is calculated based on acceptance probability. The accuracy of the analytical model is verified based on simulation results. Results from the model are compared against other models previously reported in literature. It is observed that the inaccuracy of the model measured in terms of error from simulation results is less than errors in previously reported studies.

Keywords: Acceptance Probability, Effective Memory Bandwidth, Interconnection Network, Multiple-Bus, Multiprocessor, Performance Modeling and Evaluation.

1 Introduction

Typical interconnection topologies used in shared memory multiprocessors include single bus, multiple bus, crossbar, and a variety of multi-stage interconnection networks (MIN) [3]. A multiple-bus system contains N processors $P_1, P_2, P_3, ...P_N$ (each may have its own private cache, local memory and local I/O), all connected to a set of shared memory modules by B buses $B_1, B_2, ...B_B$. The shared memory consists of M interleaved banks $M_1, M_2, ...M_M$ to allow up to M simultaneous memory accesses. The number of buses B is less than or equal to the minimum of M or N, i.e. $B \leq min(M, N)$. Each processor is connected to every bus and so is every memory bank. The special case with B equal to one, is the single-bus system, and the special case with $B = min(M, N)$ is the crossbar system.

The performance of various multiprocessor configurations have been studied widely in the past using analytical and simulation models. Basically the previous research falls into two categories: exact models [1] [15] [8] and approximate analysis [2] [6] [17] [12] [14] [13] [21]. The exact models use Petri nets, Markov chains and queuing networks. It is assumed here that if a processor's request for a memory model is rejected, that request is queued until the request is serviced. The exact models yield very accurate results but when the system becomes large, the model becomes very complicated, and the Markov chains become prohibitively complex to analyze.

Most of the work in the approximate category is based on the assumption that the requests that are not accepted in a cycle are neglected and are not resubmitted in the next cycle. Ravi [18] developed a probabilistic method to estimate the bandwidth and interference in a multiprocessor system with interleaved memory, using Stirling numbers of the second kind. Chang, Kuck and Lawrie [5] made dependency studies on processor requests and derived a very simple closed form equation. Bhuyan [2] derived results similar to Ravi's model [18], recognized Chang et al.'s [5] simple closed form equation and extended the case for systems where probability of a processor making a request is less than 1. Das and Bhuyan [6] also modeled multiprocessors systems based on the independent request assumption and arrived at analytical equations for crossbar and multiple-bus systems. But the errors in the results obtained come to around 10% in some cases, since they did not consider the dependency between requests in consecutive cycles. Liu and Jou [12] presented analytical expressions for calculating the effective memory bandwidth of multiple-bus systems. They also investigated the effect of priority on processor blocking probabilities in a system where distinct priority is assigned to each processor.

All the above analyses were based on the assumption that the requests issued in successive cycles are independent. But this assumption is unrealistic because in any practical system, a request rejected in one cycle will indeed be resubmitted in the next cycle. In fact, the large errors are only due to this simplified expression. Yen, Patel and Davidson [21] considered a model for crossbar networks in which the blocked processors resubmit their requests to the same memories in the next cycle. Their results show that this approach leads us to very accurate results. Liu and Wang [14] also considered the resubmission of rejected requests and obtained an expression for the probability of acceptance of each processor in a prioritized crossbar system. In addition they formulated a new technique to evaluate the effective memory bandwidth from the probability of acceptance. Both [21] and [14] consider crossbar systems only. In this paper, we develop a model which considers resubmission of rejected requests, for multiple-bus systems.

The objectives of this paper are (i) to develop an analytical model for prioritized multiple-bus multiprocessor systems considering resubmitted requests, and (ii) to compare the analytical model with simulation models and other analytical models previously reported in literature. In Section 2, an analytical model is developed for acceptance probability and effective memory bandwidth of multiple-bus systems. In Section 3, the analytical model is compared with simulation models and analytical models reported in literature. Section 4 offers concluding remarks.

2 An Analytical Model

Effective memory bandwidth, acceptance probability, blocking probability, throughput, etc are popular metrics used to indicate the speedup or performance of multiprocessor systems. Effective memory bandwidth (EMBW) is defined as the average number of memory modules remaining busy in a cycle. The acceptance probability of a processor is defined as the probability that the request from that processor will be accepted. The blocking probability of a processor is the complement of acceptance probability, i.e., the probability that the processor's request is blocked either due to bus or memory contention. In fact all these measures are interrelated and the one particularly favored in the past studies is EMBW. We use EMBW and acceptance probability in our performance evaluation. First we develop an analytical model for the probability of acceptance of each processor in the system. Then EMBW is computed using the acceptance probability. Stirling numbers [12] [14] [10] are very useful in evaluating the probabilities involved in a scenario like this and are described in Appendix I.

As in previous studies, our analysis is also based on the homogeneous model which assumes that each processor has the same rate R for generating new requests which are uniformly distributed among all memory modules. However, unlike other simplified models, in our analysis, the rejected requests in a cycle are not discarded, but are resubmitted in the next cycle. The other researchers did note that their assumption is unrealistic, but they assumed so in order to simplify the analysis. In our model, each processor is assigned a distinct priority, and arbitration is based on priority.

2.1 Probability of Acceptance

Let PA(k) represent the acceptance probability of processor P_k where k is the assigned priority. Processor P_0 has the highest priority and it will never be blocked, or its probability of acceptance PA(0) is always equal to 1. As k increases, the probability of being blocked increases and hence PA(k) is always smaller than $PA(k-1)$. Let E_i be the event that processor P_i does not block processor P_k for $0 \leq i \leq k$. For the cases of $k \geq 1$, PA(k) can be expressed as

$$\begin{aligned}
\text{PA(k)} &= \text{Prob}(E_0 \bigcap E_1 \bigcap E_2 \bigcap \ldots\ldots \bigcap E_{k-1}) \\
&= \text{Prob}(E_0 X Prob(E_1 \mid E_0) X Prob(E_2 \mid E_0 \bigcap E_1) \ldots. \\
&\ldots\ldots\ldots\ldots. \text{X Prob } (E_{k-1} \mid E_0 \bigcap E_1 \bigcap \ldots \bigcap E_{k-2}) \\
&= \text{PA(k-1) Prob } (E_{k-1} \mid E_0 \bigcap E_1 \bigcap \ldots \bigcap E_{k-2})
\end{aligned}$$

Processor P_0 will never be blocked. Therefore, $\text{PA}(0) = 1$. Let a_i denote the probability that processor P_i makes a new request and blocks processor P_{i+1} due to memory or bus contention, provided that processors with a priority higher than P_i do not block it, i.e., if $P_0, P_1, \ldots, P_{i-1}$ do not block P_{i+1}. Let us also define b_i as the probability that P_i resubmits a rejected request and blocks P_{i+1}. Processor P_0 is never blocked and hence never resubmits a request. Hence b_0 is always 0.

$$\begin{aligned}
\text{PA}(1) &= \text{PA}(0)\ \text{Prob}(E_0) \\
&= \text{PA}(0)\ [\ \text{Prob}(P_0 \text{ does not block } P_1)] \\
&= \text{PA}(0)\ [1 - \text{Prob}(P_0 \text{ blocks } P_1)] \\
&= \text{PA}(0)\ [1 - a_0]
\end{aligned}$$

$$\begin{aligned}
\text{PA}(2) &= PA(1) Prob(E_1 \mid E_0) \\
&= \text{PA}(1)\ [\text{conditional probability that } P_1 \\
&\qquad \text{does not block } P_2 \text{ given that} \\
&\qquad P_0 \text{ does not block } P_1] \\
&= \text{PA}(1)\ [\ 1 - \text{Prob}(\ P_1 \text{ issues a new request and} \\
&\qquad \text{blocks } P_2) - \text{Prob}(P_1 \text{ resubmits a} \\
&\qquad \text{rejected request and blocks } P_2)] \\
&= \text{PA}(1)\ [1 - a_1 - b_1]
\end{aligned}$$

⋮

In general,

$$\begin{aligned}
\text{PA(k)} &= \text{PA(k-1) [1-Prob(}P_{k-1} \text{ blocks } P_k \text{ given that} \\
&\qquad P_0, P_1, \ldots P_{k-2} \text{ do not block } P_k)] \\
&= \text{PA(k-1) [1 -(Prob that } P_{k-1} \text{ issues a new} \\
&\qquad \text{request and blocks } P_k) \text{ - (Prob that} \\
&\qquad P_{k-1} \text{ resubmits a rejected request} \\
&\qquad \text{and blocks } P_k) \text{]} \\
&= \text{PA(k-1) } [1 - a_{k-1} - b_{k-1}]
\end{aligned}$$

Derivation of a_i

Now we have to determine $a_0, a_1, \ldots a_{k-1}$ where,

$$\begin{aligned}
a_0 &= \text{Prob}(P_0 \text{ blocks } P_1) \\
a_1 &= \text{Prob}(P_1 \text{ issues a new request and blocks } P_2 \\
&\qquad \text{given that } P_0 \text{ does not block } P_2) \\
a_{k-1} &= \text{Prob}(P_{k-1} \text{ issues a new request and} \\
&\qquad \text{blocks } P_k \text{ given that } P_0, P_1, \ldots\ldots. P_{k-2} \\
&\qquad \text{do not block } P_k.)
\end{aligned}$$

Let us use A to represent the event that P_0 does not block P_2. And let B denote the event that P_1 does not block P_2 and P_0 does not block P_2. Then, a_1 is represented by the conditional probability $\text{Prob}(B \mid A)$. This conditional probability may be expressed in terms of the joint probability Prob($A \bigcap B$) and the marginal probability Prob(A). So we may write

$$\begin{aligned} a_1 &= Prob(B \mid A) \\ &= \frac{Prob(B \bigcap A)}{Prob(A)} \\ &= \frac{Prob(A) - Prob(B' \bigcap A)}{Prob(A)} \\ &= 1 - \frac{Prob(B' \bigcap A)}{Prob(A)} \\ &= 1 - \frac{Prob\ (P_0 \ \ and\ P_1 \ \ do\ not\ block\ P_2)}{Prob\ (P_0 \ \ does\ not\ block\ P_2)} \end{aligned}$$

Now redefining A and B as the events

A : P_0 and P_1 do not block P_3 and
B : P_2 blocks P_3

and proceeding in a similar fashion, we obtain

$$\begin{aligned} a_2 &= 1 - \frac{Prob(B' \bigcap A)}{Prob(A)} \\ &= 1 - \frac{Prob\ (P_0,\ P_1 \ \ and\ P_2 \ \ do\ not\ block\ P_3)}{Prob\ (P_0 \ \ and\ P_1 \ \ do\ not\ block\ P_3)} \end{aligned}$$

Similarly,

$$a_3 = 1 - \frac{Prob\ (P_0,\ P_1\ ,P_2 \ \ and\ P_3 \ \ do\ not\ block\ P_4)}{Prob\ (P_0,\ P_1 \ \ and\ P_2 \ \ do\ not\ block\ P_4)}$$

In general,

$$a_{k-1} = 1 - \frac{Prob\ (P_0, P_1, P_2 ... P_{k-1} \ \ do\ not\ block\ P_k)}{Prob\ (P_0, P_1, P_2 ... P_{k-2} \ \ do\ not\ block\ P_k)}$$

Now we have to determine the probability that $P_0, P_1, P_2, ... P_{k-1}$ do not block P_k. Bus and memory contentions have to be considered in order to determine this probability. ProcessorP_k gets blocked if the higher priority processors requested the same memory module, or they used up all the B buses in the system. As shown in [12], the probability that j memory modules are requested from (M-1) modules by i processors is

$$j!S(i,j) \begin{pmatrix} M-1 \\ j \end{pmatrix} \quad \ldots \quad (1)$$

where S(i,j) is the Stirling number of the second kind. If a request is made in every cycle, the probability that P_k is not blocked by i processors with higher priorities by using up all the B buses is

$$\frac{\sum_{j=1}^{B-1} j!S(i,j) \begin{pmatrix} M-1 \\ j \end{pmatrix}}{M^i} \quad \ldots \quad (2)$$

When the request rate of the processor is less than 1.0, we need to consider the probability of exactly i processors making a request. Hence the probability that P_k is not blocked by i higher priority processors is

$$\begin{pmatrix} k \\ i \end{pmatrix} R^i (1-R)^{k-i} \frac{\sum_{j=1}^{B-1} j!S(i,j) \begin{pmatrix} M-1 \\ j \end{pmatrix}}{M^i} \quad \ldots \quad (3)$$

Hence the probability that $P_0, P_1, P_2 \ldots P_{k-1}$ do not block P_k is

$$\sum_{i=0}^{k} \begin{pmatrix} k \\ i \end{pmatrix} R^i (1-R)^{k-i} \frac{\sum_{j=1}^{B-1} j!S(i,j) \begin{pmatrix} M-1 \\ j \end{pmatrix}}{M^i}$$

We can similarly obtain the probability that $P_0, P_1, P_2, P_3 \ldots\ldots P_{k-2}$ do not block P_k as

$$\sum_{i=0}^{k-1} \begin{pmatrix} k-1 \\ i \end{pmatrix} R^i (1-R)^{k-1-i} \ \frac{\sum_{j=1}^{B-1} j!S(i,j) \begin{pmatrix} M-1 \\ j \end{pmatrix}}{M^i}$$

Therefore,

$$a_{k-1} = 1 - \frac{\sum_{i=0}^{k} \begin{pmatrix} k \\ i \end{pmatrix} R^i (1-R)^{k-i} \frac{\sum_{j=1}^{B-1} j!S(i,j) \begin{pmatrix} M-1 \\ j \end{pmatrix}}{M^i}}{\sum_{i=0}^{k-1} \begin{pmatrix} k-1 \\ i \end{pmatrix} R^i (1-R)^{k-1-i} \frac{\sum_{j=1}^{B-1} j!S(i,j) \begin{pmatrix} M-1 \\ j \end{pmatrix}}{M^i}} \quad \ldots \quad (4)$$

This equation may be mathematically manipulated to obtain a_{k-1} as equal to $\frac{R}{M}$ for the crossbar case, and hence for crossbar if resubmitted requests are not considered,

$$PA(k) = PA(k-1) \left[1 - \frac{R}{M}\right]$$

which has also been derived in previous literature [12] using other methods.

Derivation of b_i

The probability that P_1 issues a new request and blocks P_2 is equal to a_1; hence the probability that P_1 was blocked in the previous cycle and hence resubmits a rejected request in the current cycle is

$$b_1 = a_1(1 - PA(1)).$$

Similarly, we know that a_2 is the probability that P_2 issues a new request and blocks P_3. To determine P_2's resubmission probability, we need to know the patterns in which P_2 gets blocked. P_2 could be blocked by P_0 or P_1 or both in a given cycle, but as explained in [14], while calculating P_2's resubmission probability, we need to include only the cases in which P_2 was blocked by either P_0 or by P_1 but, not by both. The probability with which P_2 resubmits a rejected request and blocks P_3 may now be given as

$$b_2 = a_2 \quad X \quad [1 - PA(2)] \quad X \frac{\begin{pmatrix} 2 \\ 1 \end{pmatrix}(a_2)(1-a_2)}{1-(1-a_2)^2} \ldots \quad (5)$$

where $\frac{\begin{pmatrix} 2 \\ 1 \end{pmatrix}(a_2)(1-a_2)}{1-(1-a_2)^2}$ represents the probability that P_2 will be blocked only by P_0 or by P_1, given that P_2 is blocked. Following the above analysis, we may obtain the expression for the general case as follows:

Prob that P_{k-1} resubmits a rejected request and blocks P_k is

$$a_{k-1} \quad X \quad [1 - PA(k-1)]$$

$$X \frac{Prob\ P_{k-1}\ is\ blocked\ by\ exactly\ one\ processor}{Prob\ P_{k-1}\ \ is\ blocked\ by\ any\ processor}$$

where a_{k-1} is the probability that P_{k-1} issues a new request and blocks P_k.

The probability that P_{k-1} is blocked by exactly one processor is, $\binom{k-1}{1} a_{k-1}(1-a_{k-1})^{k-2}$ and the probability that P_{k-1} is blocked by any processor is $1-(1-a_{k-1})^{k-1}$. Therefore,

$$b_{k-1} = a_{k-1}X[1-PA(k-1)]X\frac{\binom{k-1}{1} a_{k-1}(1-a_{k-1})^{k-2}}{1-(1-a_{k-1})^{k-1}} \tag{6}$$

As is obvious in the above explanation, term a_{k-1} considers the effect of new requests issued by higher priority processors and b_{k-1} represents the effect of rejected requests which are resubmitted. It is assumed that whenever a request is rejected, it is resubmitted until the request is granted. Until that time, obviously, the processor does not issue any new requests. In any cycle, the processor submits a maximum of one request, either a new request or a resubmitted request. In essence, we need to compensate for the fact that there are no new requests in the cycles when there is a resubmitted request. This term is properly accounted by the product of a_{k-1} and b_{k-1}, i.e., by the factor $a_{k-1}b_{k-1}$. Hence,

$$PA(k) = PA(k-1)[1 - a_{k-1} - b_{k-1} + a_{k-1}b_{k-1}] \quad \ldots \tag{7}$$

where a_{k-1} and b_{k-1} are defined as in Eq. 4 and Eq. 6. This is our analytical expression for acceptance probability.

When the term $a_{k-1}b_{k-1}$ is not considered, there are a few cycles, especially at higher request rates, in which a_{k-1} contributes a new request and b_{k-1} contributes a resubmitted request. For the special case of an N X 1 crossbar with $N = 3$, the model in [14] is reduced to

$$PA(2) = PA(1)(1 - R - R^2)$$

and this probability becomes negative when R is greater than 0.62. Using our model, since the term $a_{k-1}b_{k-1}$ is included as in the discussion above, in this special case we have,

$$PA(2) = PA(1)[1 - R - R^2 + R^3]$$

which does not become negative at any request rate.

2.2 Effective Memory Bandwidth (EMBW)

Since the effective memory bandwidth (EMBW) is the most commonly used performance index in previous research on multiprocessor systems, we also compute EMBW. This measure is defined as the expected number of accepted requests from all processors in each cycle. Acceptance probability may be used to evaluate EMBW. Since PA(k) is the acceptance probability of processor P_k, $R(k)PA(k)$ represents the contribution of the particular processor P_k towards system memory bandwidth. The effective memory bandwidth of the whole system is obtained by summing up the contributions from all processors. Hence

$$EMBW = \sum_{k=0}^{N-1} R(k)PA(k) \quad \ldots \tag{8}$$

where $R(k)$ is the request rate of processor P_k.

The request rate R is defined as the average number of requests per cycle, which is also the probability of a processor making a request in a given cycle. This request rate R does not reflect the resubmission of rejected requests. It is just the probability of making a new request in a cycle . It has been shown in Hwang [9] that an adjusted request rate considering the resubmitted requests also as new requests can be derived based on a Markov graph. This modified request rate is called dynamic request rate and is denoted by R'. This dynamic request rate of a processor with priority number k is represented by $R'(k)$. Following the derivation of dynamic request rate given in [9], it can be shown that

$$R'(k) = \frac{R}{R + PA(k)(1-R)} \quad \ldots \tag{9}$$

where $R'(k)$ is the dynamic request rate of P_k,
R is the static request rate, and
PA(k) is its probability of acceptance.

It may be noted that when request rate R is equal to 1, R' also equals 1, since Eq. 9 reduces to R/R = 1. This can be explained rather easily; a processor can submit a maximum of 1 request in a cycle, either a new request or a resubmitted request. The maximum possible dynamic request rate for any static request rate is 1.0. Considering the dynamic request rate, effective memory bandwidth becomes

$$EMBW = \sum_{k=0}^{N-1} R'(k)PA(k) \quad \ldots \tag{10}$$

where $R'(k)$ is the dynamic request rate of processor P_k.

It may be noted that the EMBW for a system of a particular size is independent of whether it is a prioritized or non-prioritized configuration. But the contributions from the individual processors are different in the two systems. In the non-prioritized system, all processors contribute equally whereas in a prioritized system, the contribution depends on the priority. Effective memory bandwidth can be calculated using Eq. 10 with probability of acceptance from Eq. 7. Equation 7 slightly overestimates PA(k) values, and hence the EMBW calculated using Eq. 10 may exceed the bus number for a configuration with only one or two buses. This problem is corrected by considering the computed value or B whichever is smaller.

3 Performance Comparison

A simulator for a prioritized multiple-bus system was developed in SIMSCRIPT II.5 [19]. A system with N processors, M memories and B buses is modeled with each processor assigned a distinct priority. The simulator accurately follows the model in Section 2, the most important feature being that the rejected requests are resubmitted in consecutive cycles. The program delivers the probability of acceptance of a request from each processor and the overall system bandwidth.

3.1 A Comparative Study

In this section the results obtained from the models in this paper are compared against models from previous researchers and simulation results.

EMBW

Analytical results on EMBW are available from Bhuyan [2], Das and Bhuyan [6], Mudge et al. [17], Liu and Jou [12], etc. In Fig 1, we plot the percent error in EMBW for 16 X 16 and 8 X 8 systems for various bus numbers, for the different models. All previous models except Mudge et al's iterative model had higher errors than the proposed model. Even Mudge's iterative model yields low errors for $R = 0.5$, but for request rate $R = 1$, the proposed model is significantly better than that model. At $R = 1$, the proposed model is significantly better than all previous models including Mudge's iterative model. At $R = 0.5$, our model is better than all models for most of the bus numbers, but Mudge's iterative model is almost as good. In the crossbar case, at R=1.0, the inaccuracy reduces from 6 or 7% to less than 0.5% for both 16 X 16 and 8 X 8 systems. The bandwidth and percentage errors for 8 X 8 X B system for various bus numbers, from our model and results from previous researchers are also presented in Table 1 and Table 2 in Appendix II.

Fig 1 illustrates the results for only request rates 0.5 and 1.0. It would be interesting to study the performance at other request rates as well. Results from [12] are available at all request rates, and hence we perform a comparison. The variation of error in EMBW with respect to request rate is illustrated in Fig 2 for 16 X 16 X 8 and 16 X 16 X 12 systems. It may be observed that the proposed model performs better than the model in [12]. It is evident from Appendix II that all models except Mudge's iterative model give similar results as in [12]. Hence the comparison in Fig 2 should hold for those models as well.

Probability of Acceptance

Previous results on acceptance probability are available from Liu and Jou [12] only. Estimated values of acceptance probability (PA(k)) values from the proposed model and the model in [12] are compared for a few representative configurations.

Figure 3 shows the percent errors for an 8 X 8 X 5 and 16 X 16 X 9 system. It may be observed that the

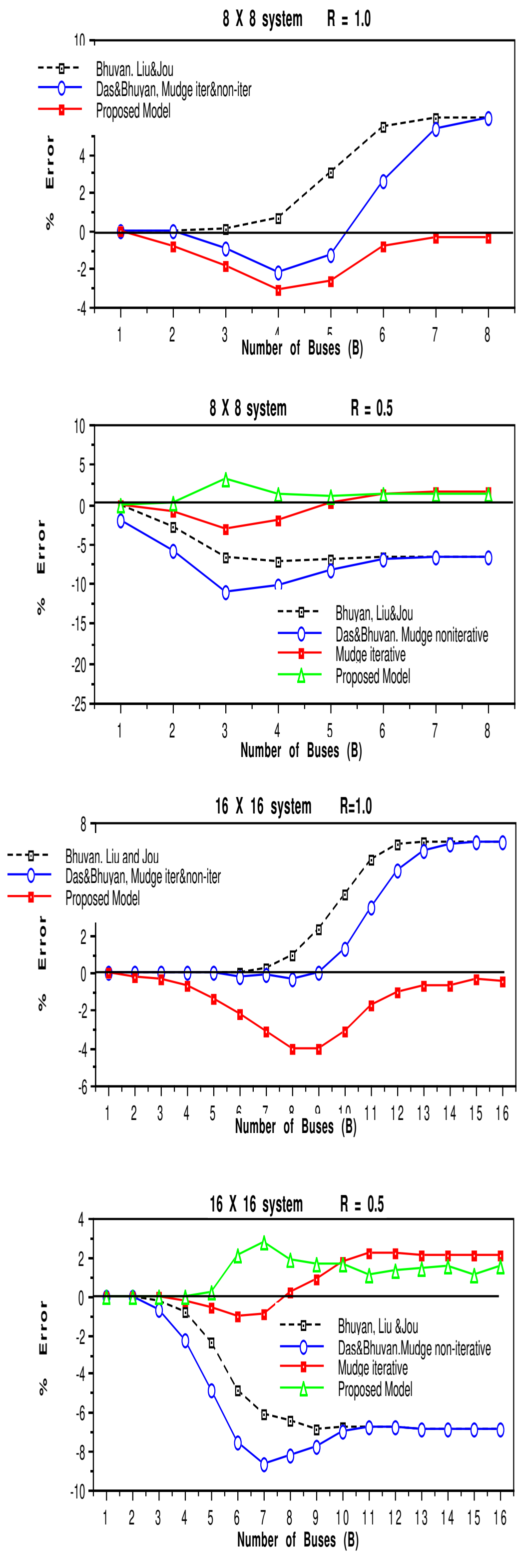

Figure 1: EMBW - Comparison of proposed model with previous studies

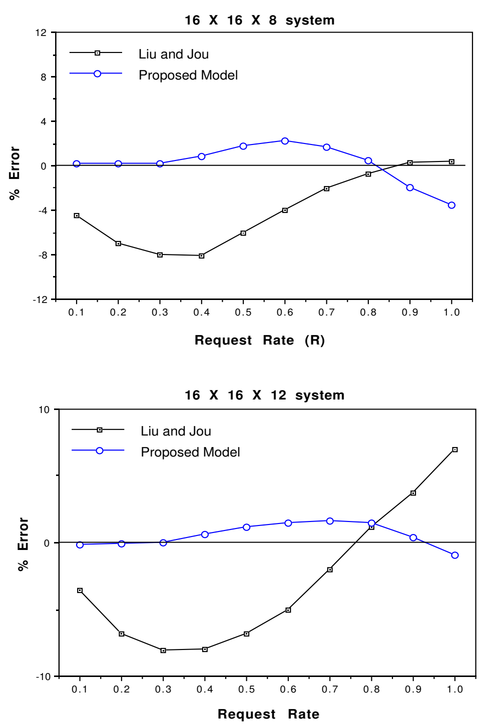

Figure 2: Error in EMBW as a function of Request rate

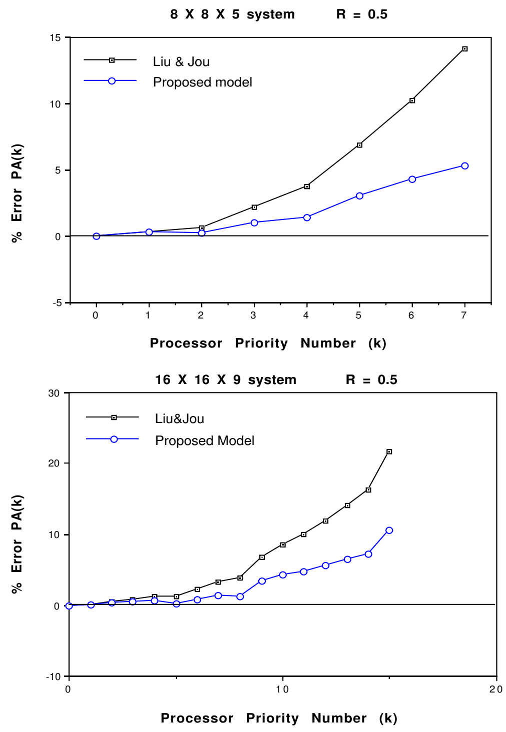

Figure 3: Percentage Error in Acceptance Probability vs Processor Priority

proposed model is significantly more accurate than the previous model in [12]. Another observation is that the model inaccuarcy increases with decrease in processor priority.

Another study is based on the PA(k) value of the lowest priority processor in the system, for various bus numbers. The lowest priority processor is chosen because that indicates the worst case performance. Fig. 4 shows the acceptance probability for the lowest priority processor in a 16 X 16 system, for $R = 0.2$ and $R = 0.5$. It is seen that the proposed model is better than the model in [12]. More detailed comparison for more cases are presented in [10].

4 Summary and Conclusion

We derived an analytical model for multiple-bus multiprocessor systems. In most of the approximate models in the past, for reason of simplicity, rejected requests are ignored. Our model considers resubmission of rejected requests and hence the results are closer to ideal (simulation) results than those reported in previous studies. Since crossbar is a special case of multiple-bus, the model can be used for crossbar systems as well.

References

[1] D. P. Bhandarkar, "Analysis of memory interference in multiprocessors", IEEE Transactions on Computers, Vol. C-24, No.9, September 1975, pp. 897-908

[2] L. N. Bhuyan, "An analysis of processor memory interconnection networks", IEEE Transactions on Computers, Vol. C-34, No. 3, March 1985, pp. 279-283.

[3] L. N. Bhuyan, Q. Yang, and D. P. Agrawal, "Performance of multiprocessor interconnection networks", Computers, Vol. 22, No.2, February 1989, pp. 25-37.

[4] L. N. Bhuyan and D. P. Agrawal, "Design and performance of generalized interconnection networks", IEEE Transactions on Computers, Vol. C-32, December 1983, pp. 1081-1090.

[5] D. Y. Chang, D. J. Kuck, and D. H. Lawrie, "On the effective bandwidth of parallel memories", IEEE Transactions on Computers, Vol. C-26, No. 5, May 1977, pp. 480-489.

[6] C. R. Das and L. N. Bhuyan, "Bandwidth availability of multiple bus multiprocessors", IEEE Transactions on Computers, Vol. C-34, No. 10, October 1985, pp. 918-926.

[7] T. Y. Feng, "A Survey of interconnection networks", IEEE Computer, December 1981, Vol. 14, No. 12, pp. 12-27.

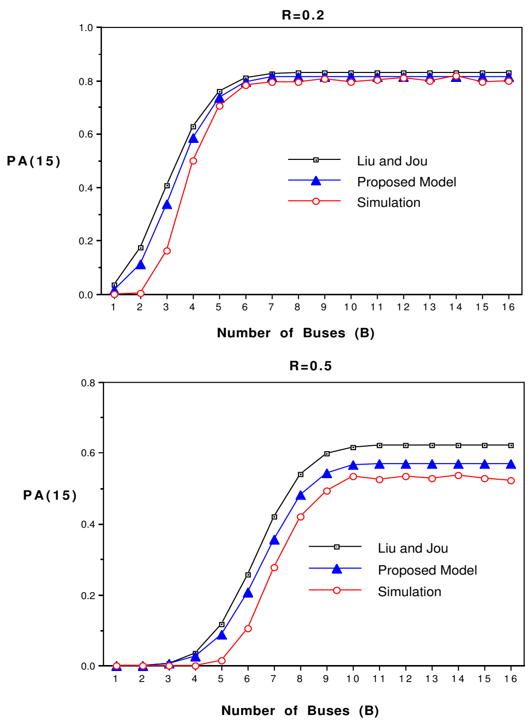

Figure 4: Acceptance Probability variation with Number of Buses

[8] M. A. Holliday and M. K. Vernon, "Exact performance estimates for multiprocessor memory and bus interference", IEEE Transactions on Computers, Vol. C-36, No. 1, January 1987, pp. 76-85.

[9] K. Hwang, and F. A. Briggs, Computer Architecture and Parallel Processing, McGraw Hill, New York, 1984.

[10] L. Kurian, Performance Evaluation of Prioritized Multiple-Bus Multiprocessor Systems, M.S. Thesis, Dept of Electrical Engr, University of Texas at El Paso, December 1989.

[11] T. Lang, M. Valero and I. Alegre, "Bandwidth of crossbar and multiple-bus connections for multiprocessors", IEEE Transactions on Computers, Vol. C-31, No.12, December 1982, pp. 1227-1234.

[12] Y. C. Liu and C. J. Jou, "Effective Memory Bandwidth and Processor Blocking Probability in multiple-bus systems", IEEE Transactions on Computers, Vol. C-36, No. 6, June 1987, pp. 761-764.

[13] Y. C. Liu and S. W. Chao, "Estimates of effective memory bandwidth for crossbar multiprocessor systems by decomposition", Proceedings of ISCA Sixth International Conference on Parallel and Distributed Computing and Systems, Louisville, Kentucky, October 1993, pp. 199-203.

[14] Y. C. Liu and C. C. Wang, "Analysis of prioritized crossbar multiprocessor Systems", Journal of Parallel and Distributed Computing, 7, October 1989, pp. 321-334.

[15] M. A. Marsan, G. Balbo, G. Conte and F. Gregoretti, "Modeling bus contention and memory interference in a multiprocessor system", IEEE Transactions on Computers, Vol. C-32, No. 1, January 1983, pp. 60-72.

[16] T. N. Mudge, J. P. Hayes, G. D. Buzzard and D. C. Winsor, "Analysis of multiple-bus interconnection Networks", Journal of Parallel and Distributed Computing, 3, March 1986, pp. 328-343.

[17] T. N. Mudge, J. P. Hayes, G. D. Buzzard and D. C. Winsor, "Analysis of multiple-bus interconnection Networks", Proceedings of the 1984 Conference on Parallel Processing, pp. 228-232.

[18] C. V. Ravi, "On the bandwidth and interference in interleaved memory systems", IEEE Transactions on Computers, Vol. C-21, No. 8, August 1972, pp. 899-901.

[19] E. C. Russel, "Building Simulation Models with SIMSCRIPT II.5, C.A.C.I., California

[20] D. Towsley, "Approximate models of multiple-bus multiprocessor systems", IEEE Transactions on Computers, Vol. C-35, No. 3, March 1986, pp. 220-228.

[21] D. W. L. Yen, J. H. Patel, and E. S. Davidson, "Memory Interference in Synchronous Multiprocessor Systems", IEEE Transactions on Computers, Vol. C-31, No. 11, November 1982, pp. 1116-1121.

Appendix I - Stirling Numbers

The number of ways of placing n distinct objects into k distinct cells with no cell left empty is equal to

$$\sum_{i=0}^{k}(-1)^i C(k,i)(k-i)^n$$

which is represented as $k!S(n.k)$ where $S(n,k)$ is called the Stirling number of the second kind and is given by

$$S(n,k) = \sum_{i=0}^{k} \frac{(-1)^i C(k,i)(k-i)^n}{k!} \qquad \ldots \qquad (11)$$

It follows that the number of ways of placing r distinct objects into n nondistinct cells with no cell left empty is equal to $S(r,n)$. The Stirling numbers of the second kind satisfy the recurrence relation

$$S(n,k) = kS(n-1,k)+S(n-1,k-1) \qquad for\ 1 < k < n-1 \qquad \ldots \qquad (12)$$

Appendix II

	Bus No.	Bhuyan	Das and Bhuyan, Mudge noniterative	Mudge iterative	Liu and Jou	Proposed	Simulation
EMBW	1	1.00	0.98	1.00	1.00	1.0000	1.0000
	2	1.94	1.88	1.98	1.94	2.0000	1.9988
	3	2.69	2.57	2.80	2.69	2.9820	2.8867
	4	3.09	2.99	3.27	3.09	3.3798	3.3346
	5	3.21	3.16	3.46	3.21	3.4889	3.4509
	6	3.23	3.22	3.51	3.23	3.5040	3.4634
	7	3.23	3.23	3.52	3.23	3.5048	3.4647
	8	3.23	3.23	3.52	3.23	3.5048	3.4647
% Error	1	0.00	-2.00	0.00	0.00	0.00	
	2	-2.94	-5.94	-0.94	-2.94	0.06	
	3	-6.81	-10.97	-3.00	-6.81	3.30	
	4	-7.34	-10.33	-1.94	-7.34	1.36	
	5	-6.98	-8.43	0.26	-6.98	1.10	
	6	-6.74	-7.03	1.35	-6.74	1.17	
	7	-6.77	-6.77	1.60	-6.77	1.16	
	8	-6.77	-6.77	1.60	-6.77	1.16	

Table 1: EMBW from various models and their percent errors (8 X 8 system, R = 0.5)

	Bus No.	Bhuyan	Das and Bhuyan, Mudge noniterative	Mudge iterative	Liu and Jou	Proposed	Simulation
EMBW	1	1.00	1.00	1.00	1.00	1.0000	1.0000
	2	2.00	2.00	2.00	2.00	1.9838	2.0000
	3	3.00	2.97	2.97	3.00	2.9435	2.9982
	4	3.98	3.87	3.87	3.98	3.8327	3.9556
	5	4.79	4.59	4.59	4.79	4.5304	4.6507
	6	5.18	5.04	5.04	5.18	4.8703	4.9116
	7	5.25	5.22	5.22	5.25	4.9356	4.9552
	8	5.25	5.25	5.25	5.25	4.9379	4.9553
% Error	1	0.00	0.00	0.00	0.00	0.00	
	2	0.00	0.00	0.00	0.00	-0.81	
	3	0.06	-0.94	-0.94	0.06	-1.82	
	4	0.62	-2.16	-2.16	0.62	-3.11	
	5	3.00	-1.31	-1.31	3.00	-2.59	
	6	5.46	2.61	2.61	5.46	-0.84	
	7	5.95	5.34	5.34	5.95	-0.40	
	8	5.95	5.95	5.95	5.95	-0.35	

Table 2: EMBW from various models and their percent errors (8 X 8 system, R = 1.0)